IN THE CLAIMS

This listing of Claims shall replace all prior versions, and listings, of claims in the application:

1. (previously presented) A boot method for synchronizing a microcontroller and a virtual microcontroller of an In-Circuit Emulation system in lock-step, comprising:

in the microcontroller, executing a set of boot code to carry out initialization;

in the virtual microcontroller, executing a set of timing code to enable a lock-step synchronization, wherein the set of timing code is a dummy code timed to take the same number of clock cycles as the microcontroller uses to execute the set of boot code, wherein the set of timing code is different from the set of boot code, and wherein the set of boot code is stored within the microcontroller and the set of boot code is inaccessible to the virtual microcontroller; and

simultaneously halting both the microcontroller and the virtual microcontroller.

2. (previously presented) The method according to Claim 1, further comprising copying register contents from the microcontroller to corresponding

registers in the virtual microcontroller after completion of the simultaneous halting.

3. (canceled)

4. (previously presented) The method according to Claim 1, wherein after the executing of the boot code, the microcontroller branches to an assembly instruction line 0; and wherein after the executing the timing code, the virtual microcontroller branches to the assembly instruction line 0.

5. (previously presented) The method according to Claim 1, wherein prior to the executing of the boot code, and prior to the executing the timing code, a break is set at an assembly instruction line 0.

6. (previously presented) The method according to Claim 1, wherein the boot code comprises protected initialization code that is not accessible to the In-Circuit Emulation system.

7. (previously presented) The method according to Claim 1, further comprising:

prior to the executing of the boot code, and prior to the executing the timing code, setting a break at an assembly instruction line 0,

wherein after the executing of the boot code the microcontroller branches to the assembly instruction line 0; and

wherein after the executing the timing code, the virtual microcontroller branches to the assembly instruction line 0.

8. (previously presented) The method according to Claim 1, further comprising:

prior to the executing of the boot code, and prior to the executing the timing code, setting a break at an assembly instruction line 0;

wherein after the executing of the boot code, the microcontroller branches to the assembly instruction line 0; and wherein after the executing the timing code, the virtual microcontroller branches to the assembly instruction line 0;

copying register contents from the microcontroller to corresponding registers in the virtual microcontroller; and

copying memory contents from the microcontroller to corresponding memory in the virtual microcontroller,

wherein after the executing of the boot code, the microcontroller branches to the assembly instruction line 0; and

wherein after the executing the timing code, the virtual microcontroller branches to the assembly instruction line 0.

9. (previously presented) The method according to Claim 8, further comprising removing the break at the assembly line 0 after the copying the register contents and the copying the memory contents.

10. (previously presented) A boot method for synchronizing a microcontroller and a virtual microcontroller of an In-Circuit Emulation system in lock-step, comprising:

resetting the microcontroller and the virtual microcontroller to a halt state;

setting a break at an assembly instruction line 0;

in the microcontroller, executing a set of boot code to carry out initialization;

in the virtual microcontroller, executing a set of timing code to enable a lock-step synchronization, wherein the set of timing code is a dummy code timed to take the same number of clock cycles as the microcontroller uses to execute the set of boot code, wherein the set of timing code is different from the set of boot code, and wherein the set of boot code is stored within the microcontroller and the set of boot code is inaccessible to the virtual microcontroller;

simultaneously halting both the microcontroller and the virtual microcontroller by branching to the assembly instruction line 0;

copying register contents from the microcontroller to corresponding registers in the virtual microcontroller;

copying memory contents from the microcontroller to corresponding memory in the virtual microcontroller; and

removing the break at the assembly line 0 after the copying the register contents and the copying the memory contents.

11. (canceled)

12. (previously presented) A boot method for synchronizing a tested device and a virtual processor of an In-Circuit Emulation system in lock-step, comprising:

in the tested device, executing a set of boot code to carry out initialization;

in the virtual processor, executing a set of timing code to enable a lock-step synchronization, wherein the timing code is a dummy code timed to take the same number of clock cycles as the tested device uses to execute the set of boot code, wherein at least one portion of the set of timing code is different from the set of boot code, and wherein the set of boot code is stored within the tested device and the set of boot code is inaccessible to the virtual processor; and

simultaneously halting both the tested device and the virtual processor.

13. (canceled)

14. (previously presented) The method according to Claim 12, further comprising copying memory contents from memory coupled to the tested device to corresponding memory coupled to the virtual processor.

15. (previously presented) The method according to Claim 12, wherein after the executing of the boot code, the tested device branches to an assembly instruction line 0; and wherein after the executing the timing code, the virtual processor branches to the assembly instruction line 0.

16. (previously presented) The method according to Claim 12, wherein prior to the executing of the boot code, and prior to the executing the timing code, a break is set at an assembly instruction line 0.

17. (previously presented) The method according to Claim 12, wherein the boot code comprises protected initialization code that is not accessible to the In-Circuit Emulation system.

18. (previously presented) The method according to Claim 12, further comprising:

prior to the executing of the boot code, and prior to the executing the timing code, setting a break at an assembly instruction line 0,

wherein after the executing of the boot code, the tested device branches to the assembly instruction line 0; and

wherein after the executing the timing code, the virtual processor branches to the assembly instruction line 0.

19. (previously presented) The method according to Claim 12, further comprising:

prior to the executing of the boot code, and prior to the executing the timing code, setting a break at an assembly instruction line 0,

wherein after the executing of the boot code, the tested device branches to the assembly instruction line 0; and wherein after the executing the timing code, the virtual processor branches to the assembly instruction line 0;

copying register contents from the tested device to corresponding registers in the virtual processor; and

copying memory contents from the tested device to corresponding memory in the virtual processor,

wherein after the executing of the boot code, the tested device branches to the assembly instruction line 0; and

wherein after the executing the timing code, the virtual processor branches to the assembly instruction line 0.

20. (previously presented) The method according to Claim 19, further comprising removing the break at the assembly instruction line 0 after the copying the register contents and the copying the memory contents.

21. (original) The method according to Claim 12, wherein the virtual processor is implemented in a field programmable gate array.

22. (previously presented) The method according to Claim 1, wherein the set of boot code comprises proprietary information, wherein the proprietary information comprises serial numbers, passwords, and algorithms.

23. (previously presented) The method according to Claim 1, wherein at least one portion of the boot code is inaccessible to the virtual microcontroller by being stored internally in the microcontroller.